TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH

KHOA CƠ KHÍ CHẾ TẠO MÁY

BỘ MÔN CƠ ĐIỆN TỬ

BÁO CÁO CUỐI KÌ

TRÍ TUỆ NHÂN TẠO

NHẬN DIỆN BỆNH COVID19 BẰNG ẢNH X- RAY

GVHD: PGS. TS Nguyễn Trường Thịnh

MHP: ARIN337629_09

Họ và tên: Phạm Huỳnh Nhật Thảo

MSSV: 20146532

Năm học: Học kì 2 2022 – 2023

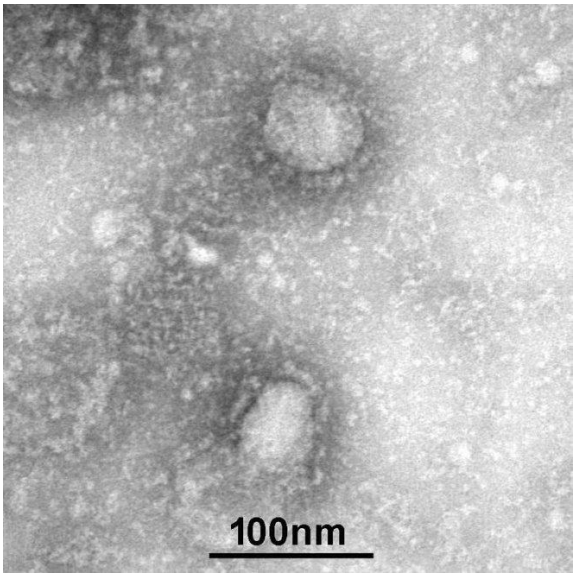
Thành phố Hồ Chí Minh, tháng 5 năm 2023

MỤC LỤC

CHƯƠNG 1. TỔNG QUAN

1.1. Giới thiệu về bệnh Covid19

Coronavirus 2019 (2019-nCoV) là virus đường hô hấp mới gây bệnh viêm đường hô hấp cấp ở người và có thể lây lan từ người sang người. Virus này được xác định trong một cuộc điều tra ổ dịch bắt nguồn từ khu chợ lớn chuyên bán hải sản và động vật ở Vũ Hán, tỉnh Hồ Bắc, Trung Quốc. 2019-nCoV là chủng virus mới chưa được xác định trước đó. Ngoài chủng coronavirus mới phát hiện này, đã có nhiều chủng coronavirus khác được biết tới ngày nay có khả năng lây nhiễm ở người với nhiều biến thể có khả năng lây nhiễm cao hơn và kháng vacxin cao hơn.



Hình 1.1. Hình ảnh Virus Covid19

Triệu chứng phổ biến nhất là: Mệt mỏi, khó thở, đau cơ xương khớp, giảm hoặc mất vị giác hoặc khứu giác, suy giảm nhận thức, rối loạn giấc ngủ, ho, đau ngực, đau đầu... Nhiều người bệnh gặp tình trạng sương mù não, mất mùi vị kéo dài, bệnh não và đột quỵ.

Vi-rút này ban đầu xuất hiện từ nguồn động vật nhưng có khả năng lây lan từ người sang người. Điều quan trọng cần lưu ý là sự lây lan từ người sang người có thể xảy ra liên tục. Ở người, virus lây từ người này sang người kia thông qua tiếp xúc với dịch cơ thể của người bệnh. Tùy thuộc vào mức độ lây lan của chủng virus, việc ho, hắt hơi hay bắt tay có thể khiến người xung quanh bị phơi nhiễm. Virus cũng có thể bị lây từ việc ai đó chạm tay vào một vật mà người bệnh chạm vào, sau đó đưa lên miệng, mũi, mắt họ. Những người

chăm sóc bệnh nhân cũng có thể bị phơi nhiễm virus khi xử lý các chất thải của người bệnh.

Các ước tính mới cho thấy tổng số ca tử vong liên quan trực tiếp hoặc gián tiếp đến đại dịch COVID-19 từ ngày 1/1/2020 đến ngày 31/12/2021 vào khoảng 14,9 triệu ca (từ 13,3 đến 16,6 triệu)". Hầu hết số ca trên (84%) được ghi nhận tại Đông Nam Á, châu Âu và châu Mỹ.. Nền kinh tế thế giới ước giảm 4,3% trong năm 2020, một mức suy giảm chỉ xảy ra trong cuộc Đại suy thoái và trong hai cuộc chiến tranh thế giới.

1.2. Lý do chọn đề tài

Như đã đề cập ở 1.1, đại dịch Covid19 là một bệnh rất nguy hiểm và ảnh hưởng trực tiếp tới sức khỏe mọi người. Ngoài ra, di chứng hậu Covid cũng là 1 vấn đề cần sự chú ý.

Một số người khỏi bệnh sau khi mắc COVID-19 nghiêm trọng có thể gặp phải các ảnh hưởng xấu tới đa cơ quan hoặc bệnh tự miễn dịch trong một thời gian dài kèm theo các triệu chứng trong nhiều tuần hoặc nhiều tháng. Không chỉ biểu hiện bằng các triệu chứng lâm sàng kể trên, người bệnh còn có thể xuất hiện những bất thường cận lâm sàng như tăng men tim kéo dài, rối loạn đường huyết, rối loạn hormon giáp, giảm độ lọc cầu thận; rối loạn chức năng hô hấp (giảm độ khuếch tán phổi, hạn chế dung tích phổi; bất thường hình ảnh học, xơ phổi, giãn phế quản trên CT scan ngực) rối loạn chức năng tâm thất qua siêu âm tim...

Các kỹ thuật học máy sâu trên X- quang ngực đang trở nên phổ biến vì chúng có thể dễ dàng sử dụng với kỹ thuật hình ảnh chi phí thấp và có nhiều dữ liệu có sẵn để đào tạo các mô hình học máy khác nhau. Đó cũng là lí do em chọn đề tài: "**Nhận diện bệnh covid19 bằng ảnh x- ray**"

1.3. Mục tiêu nghiên cứu

- Xây dựng mô hình chuẩn đoán bệnh viêm phổi bằng ảnh X-ray bằng giải pháp CNN
- Tạo GUI cho mô hình để thuận tiện cho việc sử dụng

1.4. Phương pháp nghiên cứu

- Tìm hiểu lý thuyết về dịch Covid19, tiến hành lấy dữ liệu
- Xây dựng "Model trainning" và tiến hành chạy thử để kiểm chứng

1.5. Nội dung báo cáo

Bài báo cáo tập trung vào việc nghiên cứu phương pháp tối ưu để tạo model train bằng CNN sao cho độ chính xác đạt cao nhất. Từ đó đưa ra chuẩn đoán có bị Covid19 hay không.

Bài báo cáo này được xây dựng với 4 chương, bao gồm:

CHƯƠNG 1: TỔNG QUAN

CHƯƠNG 2: CƠ SỞ LÍ THUYẾT

CHƯƠNG 3: ỨNG DỤNG

CHƯƠNG 4: KẾT LUẬN

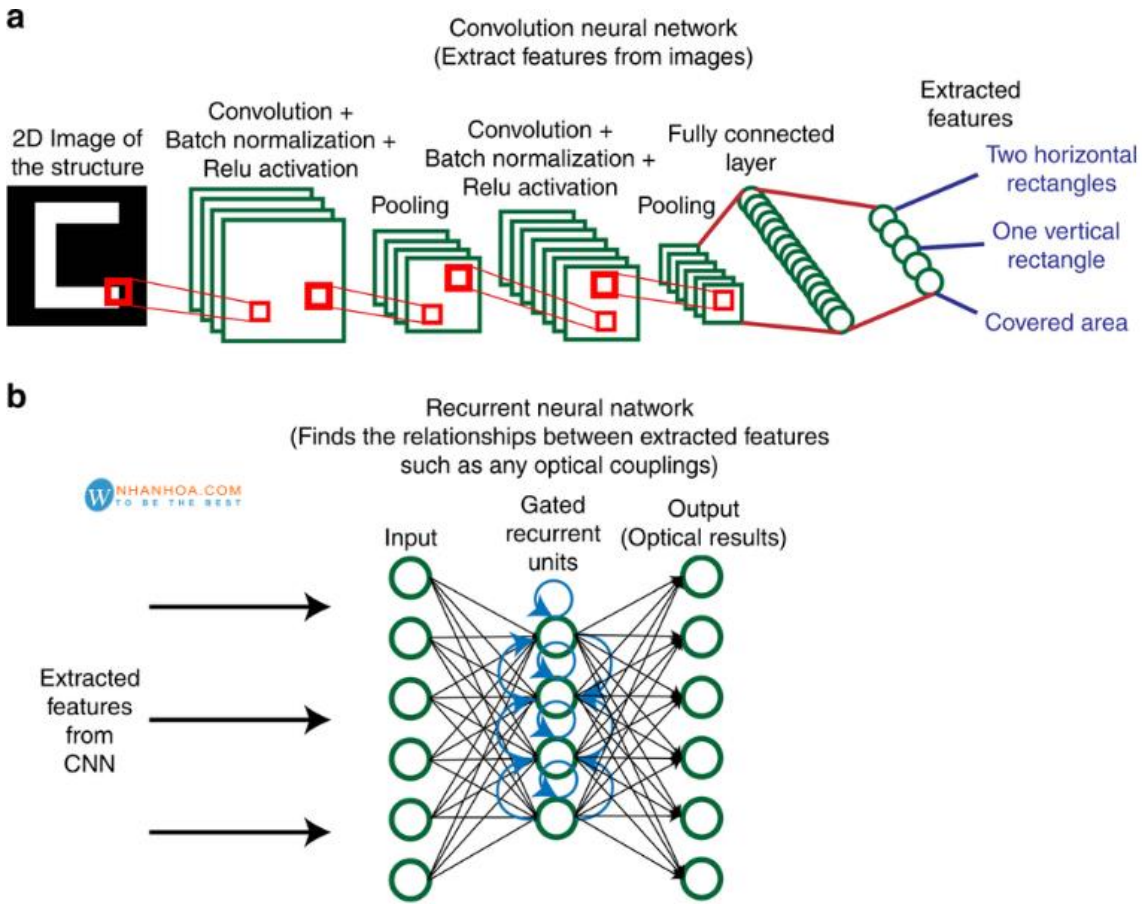
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1. Thuật toán CNN - Convolutional Neural Network

2.1.1. Convolutional Neural Network là gì

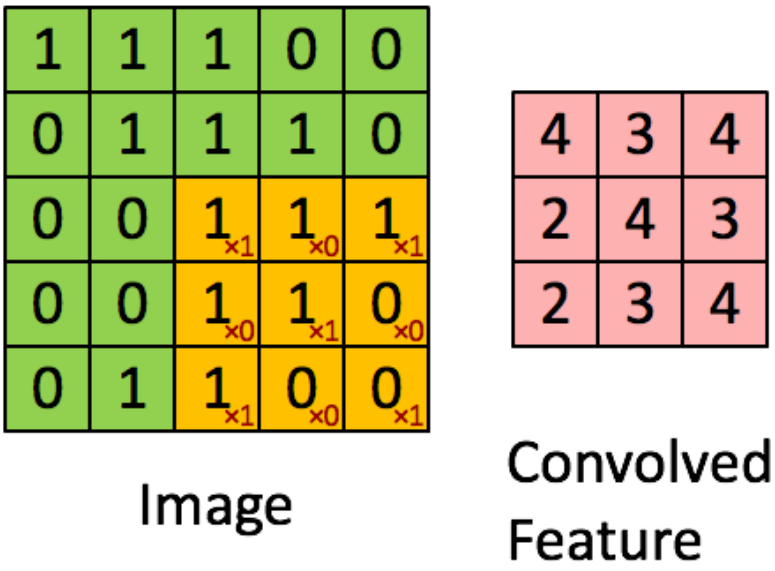
Convolutional Neural Network hay còn được viết tắt là CNN, trong tiếng Việt được gọi là mạng nơ-ron tích chập, là một trong những mô hình Deep Learning tiên tiến và hiện đại nhất hiện nay. Nhờ có CNN mà chúng ta có thể dễ dàng tạo dựng hệ thống mạng thông minh và có độ chính xác cao.

Convolutional Neural Network được sử dụng nhiều trong các bài toán nhận dạng các object trong ảnh. Để tìm hiểu tại sao thuật toán này được sử dụng rộng rãi cho việc nhận dạng (detection), chúng ta hãy cùng tìm hiểu về thuật toán này.



Hình 2.1 Convolutional Neural Network (CNN) là gì?

Convolutional là một cửa sổ trượt (Sliding Windows) trên một ma trận (xem hình)



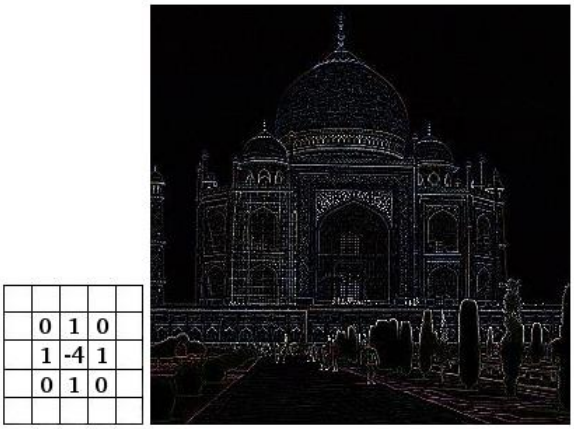
Hình 2.2. Ma trận Convolutional

Các convolutional layer có các parameter(kernel) đã được học để tự điều chỉnh lấy ra những thông tin chính xác nhất mà không cần chọn các feature.

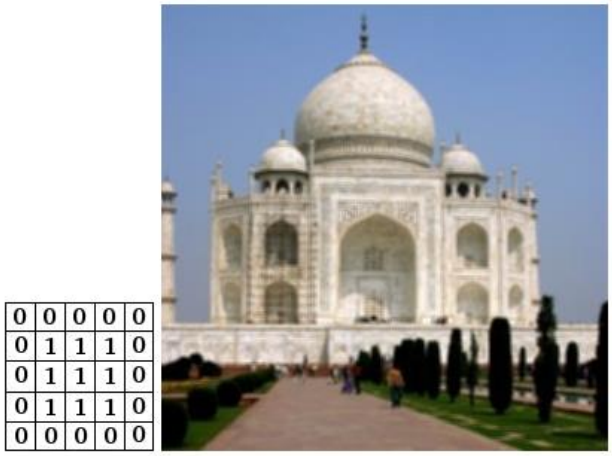
Trong hình ảnh ví dụ Hình 2.2, ma trận bên trái là một hình ảnh trắng đen được số hóa. Ma trận có kích thước 5×5 và mỗi điểm ảnh có giá trị 1 hoặc 0 là giao điểm của dòng và cột.

Convolution hay tích chập là nhân từng phần tử trong ma trận 3. Sliding Window hay còn gọi là kernel, filter hoặc feature detect là một ma trận có kích thước nhỏ như trong ví dụ trên là 3×3.

Convolution hay tích chập là nhân từng phần tử bên trong ma trận 3×3 với ma trận bên trái. Kết quả được một ma trận gọi là Convoled feature được sinh ra từ việc nhận ma trận Filter với ma trận ảnh 5×5 bên trái.



Hình 2.3. Hình ảnh trắng đen được số hóa



Hình 2.4. Hình ảnh được Convoled feature

2.1.2. Cấu trúc của mạng CNN

Mạng CNN là một tập hợp các lớp Convolution chồng lên nhau và sử dụng các hàm nonlinear activation như ReLU và tanh để kích hoạt các trọng số trong các node. Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo.

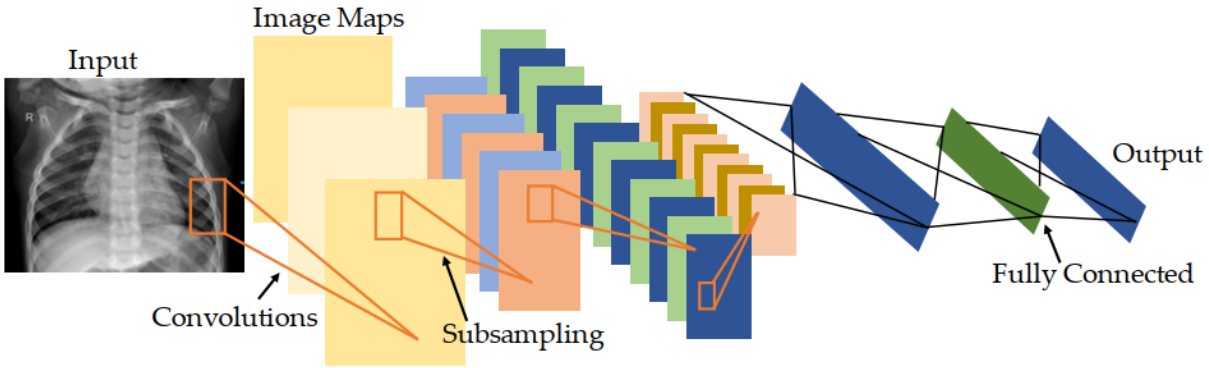
Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo. Trong mô hình mạng truyền ngược (feedforward neural network) thì mỗi neural đầu vào (input node) cho mỗi neural đầu ra trong các lớp tiếp theo.

Mô hình này gọi là mạng kết nối đầy đủ (fully connected layer) hay mạng toàn vẹn (affine layer). Còn trong mô hình CNNs thì ngược lại. Các layer liên kết được với nhau thông qua cơ chế convolution.

Layer tiếp theo là kết quả convolution từ layer trước đó, nhờ vậy mà ta có được các kết nối cục bộ. Như vậy mỗi neuron ở lớp kế tiếp sinh ra từ kết quả của filter áp đặt lên một vùng ảnh cục bộ của neuron trước đó.

Mỗi một lớp được sử dụng các filter khác nhau thông thường có hàng trăm hàng nghìn filter như vậy và kết hợp kết quả của chúng lại. Ngoài ra có một số layer khác như pooling/subsampling layer dùng để chắt lọc lại các thông tin hữu ích hơn (loại bỏ các thông tin nhiễu).

Trong quá trình huấn luyện mạng (traning) CNN tự động học các giá trị qua các lớp filter dựa vào cách thức mà bạn thực hiện. Ví dụ trong tác vụ phân lớp ảnh, CNNs sẽ cố gắng tìm ra thông số tối ưu cho các filter tương ứng theo thứ tự raw pixel > edges > shapes > facial > high-level features. Layer cuối cùng được dùng để phân lớp ảnh.



Hình 2.5. Cấu trúc mạng CNN

Trong mô hình CNN có 2 khía cạnh cần quan tâm là tính bất biến (Location Invariance) và tính kết hợp (Compositionality). Với cùng một đối tượng, nếu đối tượng này được chiếu theo các gốc độ khác nhau (translation, rotation, scaling) thì độ chính xác của thuật toán sẽ bị ảnh hưởng đáng kể.

Pooling layer sẽ cho bạn tính bất biến đối với phép dịch chuyển (translation), phép quay (rotation) và phép co giãn (scaling). Tính kết hợp cục bộ cho ta các cấp độ biểu diễn thông tin từ mức độ thấp đến mức độ cao và trừu tượng hơn thông qua convolution từ các filter. Đó là lý do tại sao CNNs cho ra mô hình với độ chính xác rất cao. Cũng giống như cách con người nhận biết các vật thể trong tự nhiên.

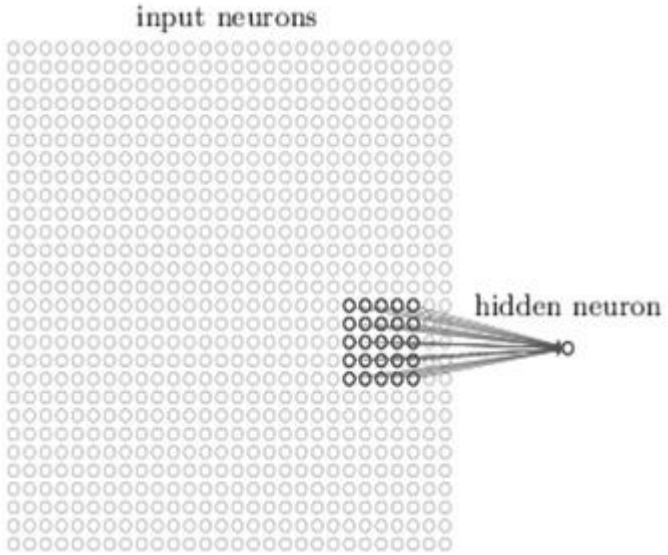
Mạng CNN sử dụng 3 ý tưởng cơ bản:

- **Các trường tiếp nhận cục bộ** (local receptive field)
- **Trọng số chia sẻ** (shared weights)
- **Tổng hợp** (pooling).

2.1.2.1. Trường tiếp nhận cục bộ (local receptive field)

Đầu vào của mạng CNN là một ảnh. Ví dụ như ảnh có kích thước 28×28 thì tương ứng đầu vào là một ma trận có 28×28 và giá trị mỗi điểm ảnh là một ô trong ma trận. Trong mô hình mạng ANN truyền thống thì chúng ta sẽ kết nối các neuron đầu vào vào tầng ảnh.

Tuy nhiên trong CNN chúng ta không làm như vậy mà chúng ta chỉ kết nối trong một vùng nhỏ của các neuron đầu vào như một filter có kích thước 5×5 tương ứng (28- 5 + 1) 24 điểm ảnh đầu vào. Mỗi một kết nối sẽ học một trọng số và mỗi neuron ẩn sẽ học một bias. Mỗi một vùng 5×5 đấy gọi là một trường tiếp nhận cục bộ.



Hình 2.6. Ma trận 28x28

Như vậy, local receptive field thích hợp cho việc phân tách dữ liệu ảnh, giúp chọn ra những vùng ảnh có giá trị nhất cho việc đánh giá phân lớp.

2.1.2.2. Trọng số chia sẻ (shared weight and bias)

Đầu tiên, các trọng số cho mỗi filter (kernel) phải giống nhau. Tất cả các nơ-ron trong lớp ẩn đầu sẽ phát hiện chính xác feature tương tự chỉ ở các vị trí khác nhau trong hình ảnh đầu vào. Chúng ta gọi việc map từ input layer sang hidden layer là một feature map. Tóm lại, một convolutional layer bao gồm các feature map khác nhau. Mỗi một

feature map giúp detect một vài feature trong bức ảnh. Lợi ích lớn nhất của trọng số chia sẻ là giảm tối đa số lượng tham số trong mạng CNN.

2.1.2.3. Lớp tổng hợp (pooling layer)

Lớp pooling thường được sử dụng ngay sau lớp convulational để đơn giản hóa thông tin đầu ra để giảm bớt số lượng neuron.

Như vậy qua lớp Max Pooling thì số lượng neuron giảm đi phân nửa. Trong một mạng CNN có nhiều Feature Map nên mỗi Feature Map chúng ta sẽ cho mỗi Max Pooling khác nhau. Chúng ta có thể thấy rằng Max Pooling là cách hỏi xem trong các đặc trưng này thì đặc trưng nào là đặc trưng nhất. Ngoài Max Pooling còn có L2 Pooling.

Cuối cùng ta đặt tất cả các lớp lại với nhau thành một CNN với đầu ra gồm các neuron với số lượng tùy bài toán.

2.2. Thư viện Tensorflow

Tensorflow là một thư viện mã nguồn mở phục vụ cho hoạt động Machine Learning. Nó được xây dựng và phát triển bởi chính Google. Trong quy trình phát triển một phần mềm bất kỳ đòi hỏi rất nhiều đoạn mã cũng như thuật toán được triển khai. Thuật toán vừa để phân tích, tổng hợp dữ liệu vừa là nền tảng để phần mềm có thể khởi chạy. Tuy nhiên chương trình càng lớn thì khối lượng phép toán càng nhiều. Cách tính toán thủ công không thể đảm bảo hiệu suất như mong muốn được. Vì thế Tensorflow xuất hiện như một chương trình hỗ trợ tính toán bằng cách tiếp cận mạnh mẽ các phép tính và bài toán phức tạp. Nhờ có Tensorflow, người dùng có thể đơn giản hóa toán học thông qua các đồ thị luồng dữ liệu tổng hợp. Ngoài ra, không thể không kể đến các ứng dụng của Tensorflow như:

- Tích hợp sẵn rất nhiều các thư viện machine learning
- Có khả năng tương thích và mở rộng tốt. Được Google phát triển cho machine learning phục vụ cả nghiên cứu lẫn xây dựng các ứng dụng thực tế

Dù đem đến nhiều lợi ích nhưng thực ra nguyên lý hoạt động của Tensorflow khá đơn giản. Về cơ bản Tensorflow sẽ giúp người dùng tạo ra các biểu đồ luồng dữ liệu hoặc những cấu trúc mô tả. Đây cũng là lý do tại sao Tensorflow được coi như là một framework. Những khung sườn này sẽ hướng dẫn dữ liệu làm cách nào để đi qua một biểu đồ hoặc một series nodes đang được xử lý. Lúc này, mỗi nodes sẽ đại diện cho một hoạt động toán học

cần xử lý. Còn mỗi kết nối hoặc mỗi edge sẽ được coi như một tensor hoặc một mảng dữ liệu đa chiều.

Như đã nêu ở phần trên, Tensorflow vốn được phát triển từ Python. Vì thế bản thân Tensorflow cũng là một ứng dụng Python. Còn các nodes và tensor trong Tensorflow là những đối tượng thuộc Python. Điều này giúp ích rất nhiều cho lập trình viên. Python vốn là một hệ thống dễ sử dụng, nó cho phép các đối tượng trừu tượng bậc cao có thể dễ dàng kết hợp với nhau. Chính nhờ sự giúp đỡ đắc lực này, quá trình phát triển phần mềm được đơn giản hóa đi rất nhiều.

2.3. Thư viện Keras

Keras chạy trên các thư viện máy mã nguồn mở như TensorFlow, Theano hoặc Bộ công cụ nhận thức (CNTK). Theano là một thư viện python được sử dụng cho các tác vụ tính toán số nhanh. TensorFlow là thư viện toán học biểu tượng nổi tiếng nhất được sử dụng để tạo mạng nơ-ron và mô hình học sâu. TensorFlow rất linh hoạt và lợi ích chính là tính toán phân tán. CNTK là khung học sâu được phát triển bởi Microsoft. Nó sử dụng các thư viện như Python, C #, C ++ hoặc các bộ công cụ học máy độc lập. Theano và TensorFlow là những thư viện rất mạnh nhưng khó hiểu để tạo mạng nơ-ron.

Keras dựa trên cấu trúc tối thiểu, cung cấp một cách dễ dàng và dễ dàng để tạo các mô hình học sâu dựa trên TensorFlow hoặc Theano. Keras được thiết kế để xác định nhanh các mô hình học sâu. Chà, Keras là một lựa chọn tối ưu cho các ứng dụng học sâu.

Keras tận dụng các kỹ thuật tối ưu hóa khác nhau để làm cho API mạng thần kinh cấp cao dễ dàng hơn và hiệu quả hơn. Nó hỗ trợ các tính năng sau:

- API nhất quán, đơn giản và có thể mở rộng.
- Cấu trúc tối thiểu - dễ dàng đạt được kết quả mà không cần rườm rà.
- Hỗ trợ nhiều nền tảng và backend.
- Thân thiện với người dùng chạy trên cả CPU và GPU.
- Khả năng mở rộng tính toán cao.

Keras năng động, mạnh mẽ và có những ưu điểm sau

- Cộng động lớn hỗ trợ
- Dễ dàng để kiểm tra.

- Mạng nơ-ron Keras được viết bằng Python giúp mọi thứ đơn giản hơn.

Keras hỗ trợ cả mạng convolution và recurrent.

2.4. Thư viện Scikit-learn (sklearn)

Scikit-learn (sklearn) là một thư viện rất mạnh mẽ, giúp mang các thuật toán học máy (machine learning) vào trong một hệ thống. Thư viện này tích hợp rất nhiều thuật toán hiện đại và cổ điển giúp bạn vừa học vừa tiến hành đưa ra các giải pháp hữu ích cho bài toán của bạn một cách đơn giản. Thư viện cung cấp một tập các công cụ xử lý các bài toán machine learning và statistical modeling gồm: classification, regression, clustering, và dimensionality reduction.

Thư viện được cấp phép bản quyền chuẩn FreeBSD và chạy được trên nhiều nền tảng Linux. Scikit-learn được sử dụng như một tài liệu để học tập.

Để cài đặt scikit-learn trước tiên phải cài thư viện SciPy (Scientific Python). Những thành phần gồm:

- Numpy: Gói thư viện xử lý dãy số và ma trận nhiều chiều
- SciPy: Gói các hàm tính toán logic khoa học
- Matplotlib: Biểu diễn dữ liệu dưới dạng đồ thị 2 chiều, 3 chiều
- IPython: Notebook dùng để tương tác trực quan với Python
- SymPy: Gói thư viện các kí tự toán học
- Pandas: Xử lý, phân tích dữ liệu dưới dạng bảng

Những thư viện mở rộng của SciPy thường được đặt tên dạng SciKits. Như thư viện này là gói các lớp, hàm sử dụng trong thuật toán học máy thì được đặt tên là scikit-learn.

Scikit-learn hỗ trợ mạnh mẽ trong việc xây dựng các sản phẩm. Nghĩa là thư viện này tập trung sâu trong việc xây dựng các yếu tố: dễ sử dụng, dễ code, dễ tham khảo, dễ làm việc, hiệu quả cao.

Mặc dù được viết cho Python nhưng thực ra các thư viện nền tảng của scikit-learn lại được viết dưới các thư viện của C để tăng hiệu suất làm việc. Ví dụ như: Numpy (Tính toán ma trận), LAPACK, LibSVM và Cython.

Thư viện tập trung vào việc mô hình hóa dữ liệu. Nó không tập trung vào việc truyền tải dữ liệu, biến đổi hay tổng hợp dữ liệu. Những công việc này dành cho thư viện Numpy và Pandas.

CHƯƠNG 3. ỨNG DỤNG

3.1. Xây dựng mô hình – Trainning model

Sử dụng Google Colab để xây dựng mô hình trainning.

Bước 1: Liên kết Google Drive

```
from google.colab import drive
drive.mount('/content/drive')
```

Bước 2: Khai báo các thư viện cần thiết

```
from keras.models import Sequential, Model
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D,
Normalization
from keras.layers import LeakyReLU
from keras.optimizers import Adam
from os import listdir
from numpy import asarray, save
from keras.utils import load_img
from keras.utils import img_to_array
import numpy as np
from numpy import asarray
from keras.preprocessing.image import ImageDataGenerator
```

Bước 3: Lấy dữ liệu và tăng cường dữ liệu (data augmentation)

Augmentation là kĩ thuật tạo ra dữ liệu training từ dữ liệu mà ta đang có. Việc này giúp chúng ta tạo ra nhiều hình ảnh hơn từ hình ảnh gốc.

```
train_data=ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2,
horizontal_flip=True)
train_datagen = ImageDataGenerator(rescale = 1./255,shear_range = 0.2,
zoom_range = 0.2,horizontal_flip = True)
training =
train_datagen.flow_from_directory('/content/drive/MyDrive/data/train',
target_size = (150,150), batch_size = 32,class_mode='categorical')
validation=train_data.flow_from_directory('/content/drive/MyDrive/data/valid',
target_size=(150,150), batch_size=32, class_mode='categorical')
```

```
Found 1650 images belonging to 2 classes.
Found 640 images belonging to 2 classes.
```

Trong phạm vi dữ liệu em có, em tiến hành tăng cường dữ liệu:

- **zoom_range**: thực hiện zoom ngẫu nhiên trong phạm vi 0.2

- **rescale**: Scale lại với tỉ lệ 1/255
- **rotation_range**: Xoay ảnh góc 0.2 độ

Kết quả ta được train_set có 1650 hình ảnh thuộc 2 lớp, val_set có 640 hình ảnh thuộc 2 lớp.

Bước 4: Xây dựng mô hình CNN

```
model=Sequential()

model.add(Conv2D(32,(3,3), activation='relu',
kernel_initializer='he_uniform',
padding='same',input_shape=(150,150,3)))
model.add(Conv2D(32,(3,3), activation='relu',
kernel_initializer='he_uniform', padding='same'))

model.add(MaxPooling2D(2,2))
model.add(Conv2D(64,(3,3), activation='relu',
kernel_initializer='he_uniform', padding='same'))
model.add(Conv2D(64,(3,3), activation='relu',
kernel_initializer='he_uniform', padding='same'))

model.add(MaxPooling2D(2,2))
model.add(Conv2D(128,(3,3), activation='relu',
kernel_initializer='he_uniform', padding='same'))
model.add(Conv2D(128,(3,3), activation='relu',
kernel_initializer='he_uniform', padding='same'))

model.add(MaxPooling2D(2,2))
model.add(Flatten())
model.add(Dense(256,activation='relu',kernel_initializer='he_uniform'))
model.add(Dropout(0.2))
model.add(Dense(2,activation='softmax'))
model.compile(
    optimizer = 'Adam',
    loss='categorical_crossentropy',
    metrics = 'accuracy'
)

model.summary()
```

Cấu trúc mô hình CNN được sử dụng trong đề tài cụ thể như sau:

- Lớp đầu tiên là lớp Convolution2D với dữ liệu đầu vào có kích thước là (32x32x3), Lớp này có kích thước cửa sổ trượt (Kernel_size) là (3,3) và bước trượt là (2,2).

- Tiếp theo là lớp Maxpooling2D, kích thước (pool_size) là (2,2), lớp này có nhiệm vụ giảm kích thước ảnh tuy nhiên vẫn giữ nguyên các đặc trưng của ảnh, từ đó không làm thay đổi đặc tính dữ liệu.
- Lớp thứ ba là lớp Convolution2D với bộ lọc (filter) là 64, kích thước cửa sổ trượt (kernel_size) là (3,3), kích thước trượt là (2,2)
- Lớp thứ tư là lớp Convolution2D với bộ lọc (filter) là 64, kích thước cửa sổ trượt (kernel_size) là (3,3), kích thước trượt là (2,2)
- Lớp thứ năm là lớp Convolution2D với bộ lọc (filter) là 128, kích thước cửa sổ trượt (kernel_size) là (2,2), kích thước trượt là (1,1)
- Lớp thứ sáu là lớp Maxpooling2D với kích thước pool_size là (2,2)
- Tiếp theo là lớp Falten, lớp này có nhiệm vụ đưa tất cả dữ liệu về kích thước (1,1,Chiều sâu) hay nói cách khác, dữ liệu khi qua lớp này sẽ được trải phẳng và có dạng một chiều.

Các thông số Input, Ouput cụ thể được thể hiện trong bảng thông số mô hình dưới dây:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 150, 150, 32)      896

 conv2d_1 (Conv2D)           (None, 150, 150, 32)      9248

 max_pooling2d (MaxPooling2D  (None, 75, 75, 32)       0
 )

 conv2d_2 (Conv2D)           (None, 75, 75, 64)        18496

 conv2d_3 (Conv2D)           (None, 75, 75, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 37, 37, 64)       0
 2D)

 conv2d_4 (Conv2D)           (None, 37, 37, 128)       73856

 conv2d_5 (Conv2D)           (None, 37, 37, 128)       147584

 max_pooling2d_2 (MaxPooling  (None, 18, 18, 128)      0
 2D)

 flatten (Flatten)           (None, 41472)             0

 dense (Dense)               (None, 256)               10617088

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 2)                 514
```

```
=================================================================
Total params: 10,904,610
Trainable params: 10,904,610
Non-trainable params: 0
_________________________________________________________________
```

Bước 5: Huấn luyện mô hình

```
train = model.fit(training,epochs =
50,validation_data=validation,verbose = 1)
```

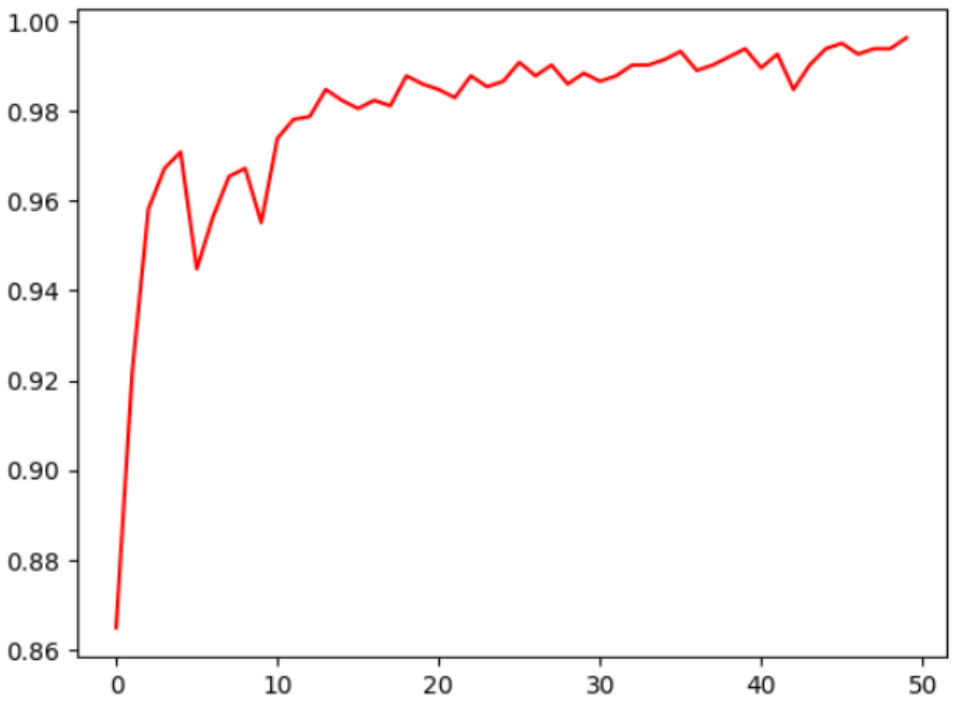
Mô hình được huấn luyện với số lần lọc 50, ta được kết quả như sau:

```
Epoch 1/50
52/52 [==============================] - 649s 12s/step - loss: 0.7733 -
accuracy: 0.8648 - val_loss: 0.3317 - val_accuracy: 0.9281
Epoch 2/50
52/52 [==============================] - 64s 1s/step - loss: 0.1991 -
accuracy: 0.9218 - val_loss: 0.2338 - val_accuracy: 0.9078
Epoch 3/50
52/52 [==============================] - 62s 1s/step - loss: 0.1124 -
accuracy: 0.9582 - val_loss: 0.1482 - val_accuracy: 0.9500
Epoch 4/50
52/52 [==============================] - 64s 1s/step - loss: 0.0948 -
accuracy: 0.9673 - val_loss: 0.1403 - val_accuracy: 0.9422
Epoch 5/50
52/52 [==============================] - 63s 1s/step - loss: 0.0797 -
accuracy: 0.9709 - val_loss: 0.1556 - val_accuracy: 0.9453
Epoch 6/50
52/52 [==============================] - 64s 1s/step - loss: 0.1662 -
accuracy: 0.9448 - val_loss: 0.1587 - val_accuracy: 0.9453
Epoch 7/50
52/52 [==============================] - 63s 1s/step - loss: 0.1121 -
accuracy: 0.9564 - val_loss: 0.1754 - val_accuracy: 0.9359
Epoch 8/50
52/52 [==============================] - 64s 1s/step - loss: 0.0909 -
accuracy: 0.9655 - val_loss: 0.1368 - val_accuracy: 0.9469
Epoch 9/50
52/52 [==============================] - 71s 1s/step - loss: 0.0778 -
accuracy: 0.9673 - val_loss: 0.1533 - val_accuracy: 0.9516
Epoch 10/50
52/52 [==============================] - 64s 1s/step - loss: 0.1379 -
accuracy: 0.9552 - val_loss: 0.1497 - val_accuracy: 0.9391
Epoch 11/50
52/52 [==============================] - 63s 1s/step - loss: 0.0900 -
accuracy: 0.9739 - val_loss: 0.1029 - val_accuracy: 0.9578
Epoch 12/50
52/52 [==============================] - 65s 1s/step - loss: 0.0624 -
accuracy: 0.9782 - val_loss: 0.1984 - val_accuracy: 0.9281
Epoch 13/50
52/52 [==============================] - 65s 1s/step - loss: 0.0610 -
accuracy: 0.9788 - val_loss: 0.1610 - val_accuracy: 0.9453
```

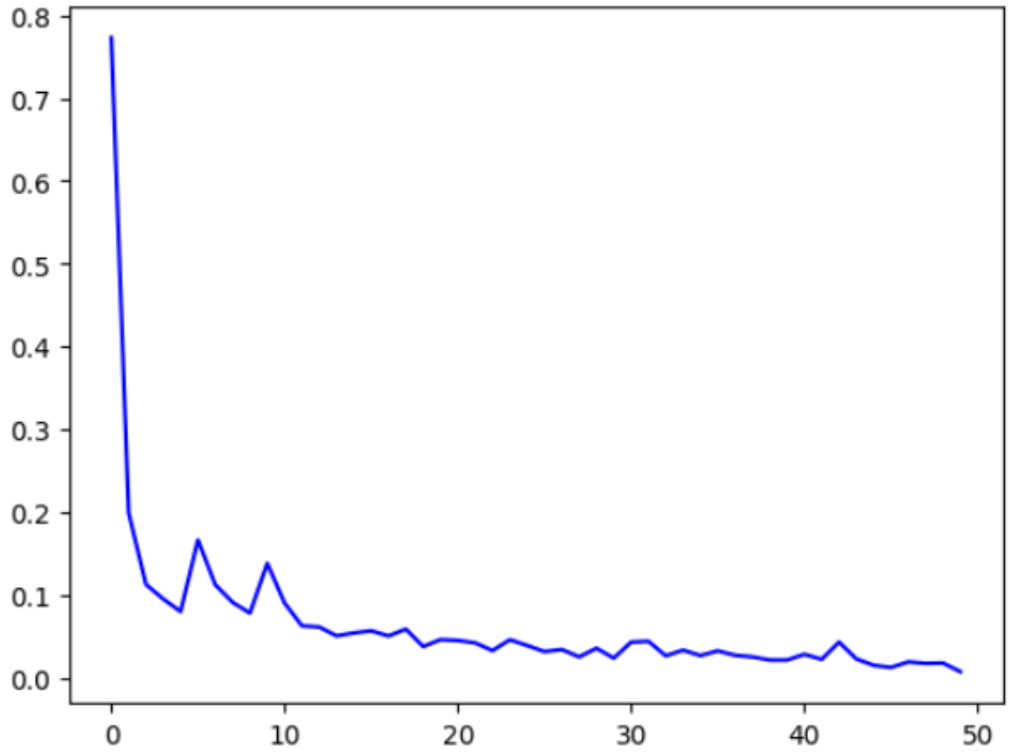
```
Epoch 48/50
52/52 [==============================] - 64s 1s/step - loss: 0.0170 -
accuracy: 0.9939 - val_loss: 0.1131 - val_accuracy: 0.9688
Epoch 49/50
52/52 [==============================] - 64s 1s/step - loss: 0.0175 -
accuracy: 0.9939 - val_loss: 0.0790 - val_accuracy: 0.9766
Epoch 50/50
52/52 [==============================] - 64s 1s/step - loss: 0.0071 -
accuracy: 0.9964 - val_loss: 0.0734 - val_accuracy: 0.9781
```

Từ kết quả mô hình huấn luyện, ta có được đồ thị độ chính xác theo số lần lọc:

```
# Vẽ đồ thị
import matplotlib.pyplot as plt
accuracy = train.history['accuracy']
loss = train.history['loss']
epochs = range(len(accuracy))
# Vẽ đồ thị giữa số lần học (Epochs) và độ chính xác (Accuracy)
plt.plot( epochs, accuracy, 'r', label = 'Training accuracy')
plt.show()
# Vẽ đồ thị giữa số lần học (Epochs) và độ mất mác (loss)
plt.plot( epochs, loss, 'b', label = 'Training loss')
plt.show()
```



Hình 3.1. Đồ thị độ chính xác của mô hình theo số lần học



Hình 3.2. Đồ thị giá trị mất mát theo số lần học

3.2. Xây dựng GUI với Python

```
import tkinter as tk
from tkinter import *
from PIL import ImageTk, Image
import numpy as np
from tensorflow import keras
# Load Model
model = keras.models.load_model('covid1.h5')
# Create the GUI window
from tkinter import filedialog

window = tk.Tk()
window.title("Covid Prediction")
window.geometry("400x400")

# Create a label to display the prediction result
result_label = tk.Label(window, text="Prediction: ")
result_label.pack()


# Function to perform covid19 prediction
def predict_disease(image_path):
    img = Image.open(image_path).resize((150, 150))
    img_array = np.array(img) / 255.0
    img_array = np.expand_dims(img_array, axis=0)

    # Make prediction using the loaded model
    prediction = model.predict(img_array)
    # Assuming the model outputs one-hot encoded labels, convert prediction
to class labels
    class_labels = ['Covid', 'Normal']
    predicted_label = class_labels[np.argmax(prediction)]
```

```
    result_label.config(text="Prediction: " + predicted_label)


# Function to handle button click event
def open_image():
    image_path = filedialog.askopenfilename(initialdir="test_images",
title="Select Image",
                                            filetypes=(("JPEG files",
"*.jpg"), ("PNG files", "*.png")))
    predict_disease(image_path)

    # Display the selected image in the GUI
    img = Image.open(image_path).resize((150, 150))
    img_tk = ImageTk.PhotoImage(img)
    img_label = tk.Label(window, image=img_tk)
    img_label.image = img_tk
    img_label.pack()
# Create a button to open an image for prediction
open_button = tk.Button(window, text="Open Image", command=open_image)
open_button.pack()


window.mainloop()
```

CHƯƠNG 4. KẾT LUẬN

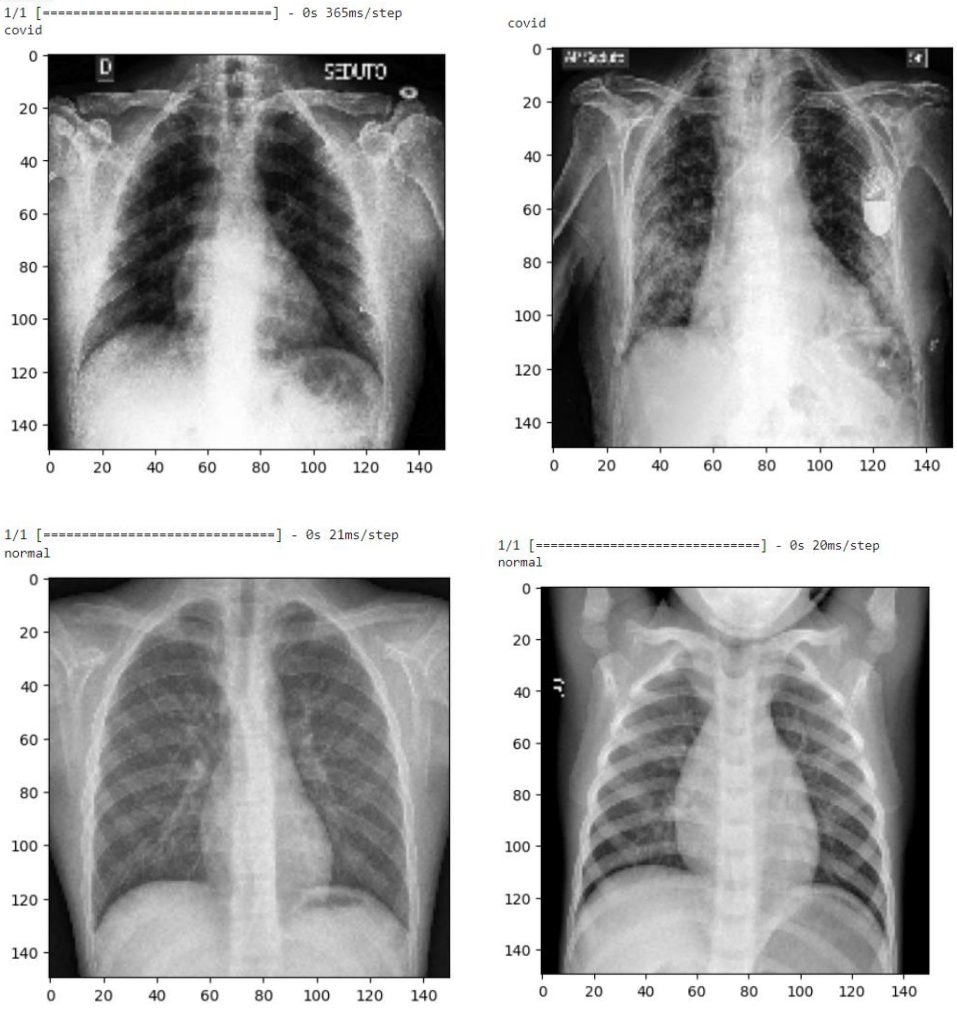
4.1. Kết quả đạt được

4.1.1. Mô hình chuẩn đoán

Mô hình CNN chuẩn đoán bệnh viêm phổi với độ chính xác 99%, cho ra kết quả chuẩn đoán khá chính xác, phân biệt được phổi không bị bệnh, phổi bị bệnh covid19

Sử dụng google colab để chạy dự đoán, chúng ta có một số kết quả sau:

```
test="/content/drive/MyDrive/data/test"
for i in listdir(test):
  img=load_img(test+'/'+i,target_size=(150,150))
  plt.imshow(img)
  img=img_to_array(img)
  img=img.astype('float32')
  img=img/255
  img=np.expand_dims(img,axis=0)
  result=(model_1.predict(img).argmax())
  class_name=['covid','normal']
  print(class_name[result])
  plt.show()
```

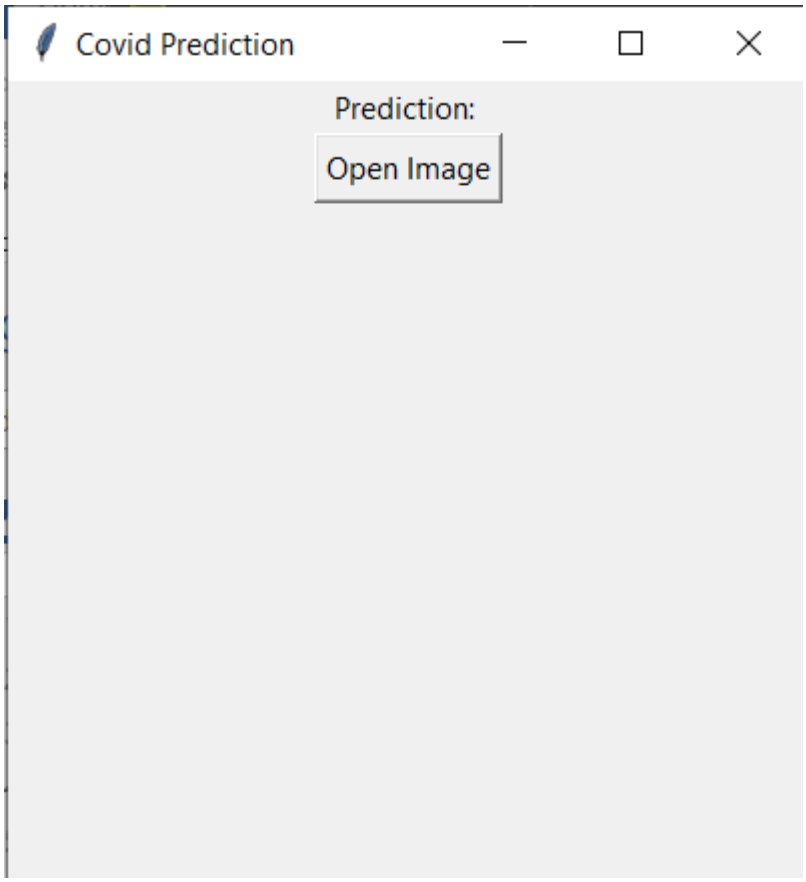


Hình 4.1. Hình chuẩn đoán người bị bệnh và người bình thường

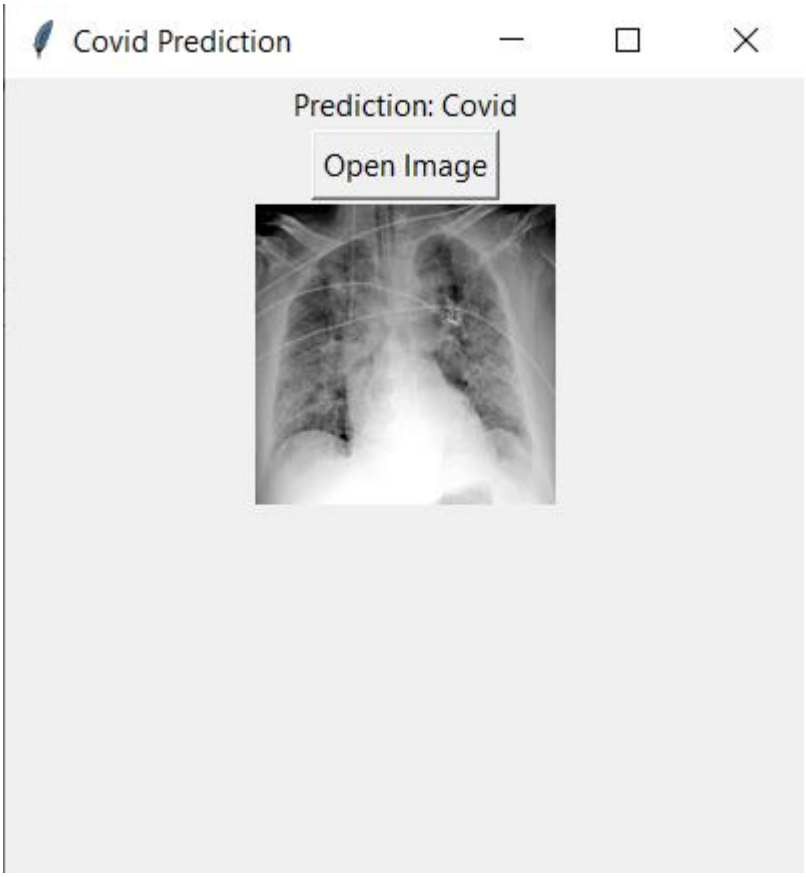
4.1.2. GUI

Chương trình được viết trên ngôn ngữ Python

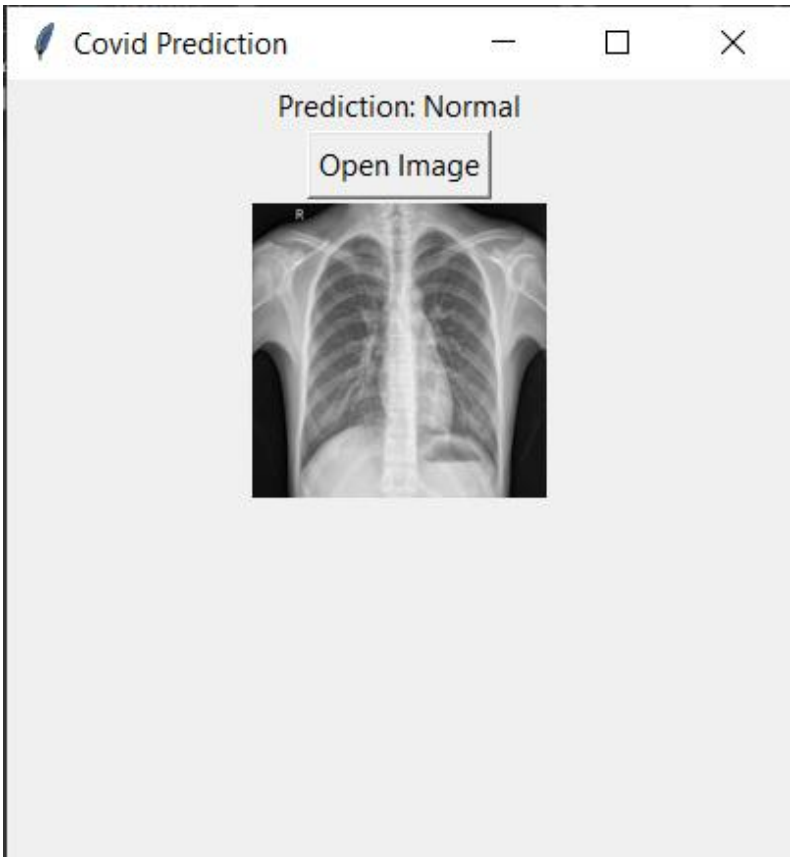
Một số hình ảnh được đưa vào chuẩn đoán:



Hình 4.2. Hình ảnh lúc chưa chuẩn đoán



Hình 4.3. Hình ảnh lúc chuẩn đoán mắc Covid19



Hình 4.3. Hình ảnh lúc chuẩn đoán phổi người bình thường

4.2. Nhược điểm

Mô hình tạo ra với độ chính xác tương đối cao nhưng vẫn còn chuẩn đoán sai một số hình ảnh.

Mô hình chưa kết hợp với chạy thời gian thực nên tính ứng dụng chưa cao.

4.3. Hướng phát triển

Trong tương lai, em sẽ tiếp tục phát triển huấn luyện mô hình để nâng độ chính xác lên cao hơn. Bên cạnh đó, em sẽ tiếp tục hoàn thiện GUI, kết hợp với máy chụp X-rays để chuẩn đoán trực tiếp.

TÀI LIỆU THAM KHẢO

[1]. Vinmec.com, Thông tin đầy đủ về dịch virus Corona 2019 theo hướng dẫn của Bộ Y tế

Link truy cập: https://www.vinmec.com/vi/tin-tuc/thong-tin-suc-khoe/dich-2019-ncov/thong-tin-suc-khoe/thong-tin-day-du-ve-virus-corona-theo-huong-dan-cua-bo-y-te/

[2]. Báo sức khỏe và đời sống, Làm sao tính được thiệt hại mà COVID-19 gây ra cho toàn nhân loại?

Link truy cập: https://suckhoedoisong.vn/tinh-toan-thiet-hai-ma-covid-19-gay-ra-cho-toan-nhan-loai-bang-cach-nao-1692205231835078.htm

[3]. TOPDev, Thuật toán CNN – Convolutional Neural Network, Link truy cập: https://topdev.vn/blog/thuat-toan-cnn-convolutional-neural-network/

PHỤ LỤC

1. Link Datasets: https://www.kaggle.com/datasets/jtiptj/chest-xray-pneumoniacovid19tuberculosis
2. Link Github: https://github.com/utopickaiser/Final-Project-AI/
3. Link Youtube: https://youtu.be/NhHn3bKtgU8